# A Microcontroller-based HF-RFID Reader Implementation for the SD-Slot

Andreas Loeffler, Andreas Deisinger
*Chair of Information Technologies*
*Friedrich-Alexander-University of Erlangen-Nuremberg*
*Erlangen, Germany*
*Email: loeffler@like.eei.uni-erlangen.de, Andreas.Deisinger@e-technik.stud.uni-erlangen.de*

***Abstract*—This work describes an RFID reader system based on an emulated file system to be used in SD-capable systems. Off-the-shelf SD-compliant HF-RFID readers, generally, use the SDIO interface for connecting to computers and PDAs. Therefore, the usage of SDIO-compatible SD card readers is essential to assure correct operation. In contrast to the adoption of SDIO, this work shows an approach to be used with every SD card reader, not necessarily requiring the SDIO specification. This leads to an HF-RFID system to be applied in computer environments (PDA, PC, etc.) where independence of operating systems and special drivers is of great importance. Adding RFID functionalities to existent systems could help to minimize the gap between real and digital world. This approach offers this particular functionality to any SD-compliant host device.**

***Keywords*-Radiofrequency identification, file systems, microcontrollers, smart cards, emulation.**

## I. Introduction

The market for RFID applications is still expanding [1]. Therefore, the need for RFID readers [2], particularly mobile RFID readers, is increasing.

There are plenty of RFID readers for nearly every possible kind of interface, like Ethernet [3], SPI, serial port, Bluetooth, USB, etc. Interfaces mainly used by mobile devices are, apart from wireless interfaces, USB, SecureDigital (SD), and some proprietary interfaces. The USB interface on mobile devices is usually driven in device mode not in host mode. Moreover, there exist various USB socket implementations and particular derivatives. These are some reasons why various RFID readers are connected to (primarily mobile) computers and PDAs using the SD interface [4]. Unfortunately, common RFID reader implementations (e.g., [5], [6]) usually prefer the SD Input/Output (SDIO)-interface. Using SDIO instead of standard SD has several disadvantages. The first drawback is the need for an SDIO-compatible SD card reader. The second disadvantage is the need for various drivers to be installed using such an SDIO-capable RFID reader. As some readers only support drivers for one specific operating system, there is rather no reason in upgrading existing systems with RFID functionalities. Besides, the costs of such SDIO readers are usually higher compared to ordinary RFID readers.

The subject of this paper describes an approach to realize an SD-capable (not SDIO) HF-RFID reader to be used in every ordinary SD card reader. It is important to outline the advantages of the implementation. There are two major issues to be regarded. First, full operating system independence, and, second, no additional drivers need to be loaded as the computer's operating system will recognize the reader as an emulated SD card.

This paper will focus on the hardware part and the realization of such a system and is organized as follows. Section II gives a short description of the system. Section III shows the verification and Section IV the limitations of the current implementation of the system. A conclusion and references for future work are given in Section V.

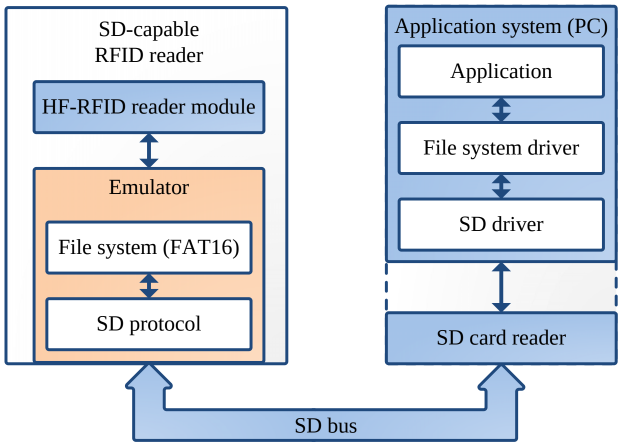


Figure 1. The block diagram of the whole system, including the SD-capable reader (left block) linked via SD bus to the application system (right block, e.g., a PC or PDA)

## II. System

This section will provide an overview of the system as a whole. Figure 1 shows the concept of the SD-capable HF-RFID reader. On the left side, there is the current SD-capable RFID reader prototype connecting several blocks including one block for the emulation of the SD card protocol and one block for emulating a File Allocation Table (FAT)-16 file system [7]. The content of the file system is represented by mapped objects; these objects include links to the RFID reader's firmware itself, providing the RFID transponders' data (e.g., the Unique Identifiers (UIDs) of several transponders within the read range of the RFID reader).

The right side of Figure 1 is represented by an ordinary SD card reader, either externally connected, e.g., via USB, or an internal one. The goal of this system is to provide the SD-capable application system (i.e., a PC, PDA or smartphone) with the data of the RFID transponders read out. This is realized by creating a file within the emulation of the FAT file system, which is subsequently read out by the application system. This file includes the data of the transponders, i.e., the output of the RFID reader. This is also known as RFID uplink channel. The downlink channel, i.e., data from the reader to the tags, may also be realized using a file-based approach. By writing information into an emulated file, the SD-capable reader may notice the change and forward the commands to the underlying RFID reader hardware. Because of simplicity reasons, the system described in this work exclusively provides an RFID uplink channel, i.e., the connection from the the RFID transponders to the SD card reader or application, respectively.

## A. *Hardware*

This subsection describes the hardware of the SD-capable RFID reader. Figure 2 shows the basic blocks the reader is built upon. Peripheral parts include a *Debugging interface*, an *RFID interface* to connect to an earlier developed HF-RFID reader module, a *Programming interface* to be able to program the microcontroller (µC) of type Atmel AT32UC3A1512 [8] using either the JTAG interface or USB, and a power supply module providing primarily the µC with 3.3 V.

The main part of the hardware is a 32 bit µC of the AT32UC3A family from Atmel. The main task of the device is the communication with the SD card reader connected over the SD bus. Figure 3 shows the prototype of the SD-capable HF-RFID reader. The microcontroller is in the center of the figure as well as the SD interface (right hand side) that may be connected to an SD card reader. The RFID interface for connecting the RFID module is shown at the bottom of the figure, whereas the JTAG interface serves as programming and debugging interface for the µC's firmware. The USB and RS-232 connections at the top are used as programming (USB) and system debugging (RS-232) sources. System debugging includes the retrieval of system status and error messages, which are generated by the firmware of the µC.

## B. *Firmware*

The next two paragraphs give further details of the SD protocol and its realization within the reader.

*1) Basics of the SD Protocol:* The SD protocol is a Master-Slave protocol, which means that every step during communication is triggered by the master (SD card reader) followed by the slave's (SD-capable HF-RFID reader) response. After inserting the RFID reader into an SD card reader, the card reader tries to access the *emulated* card. The SD card reader checks the type of card inserted. This could be either an SD or a Multimedia Card (MMC) [9], a predecessor of the SD card. The communication speed at this point has a maximum clock rate of 400 kHz (from the SD card reader), which is defined by the standard. So far, only one communication line on the SD bus is used. The firmware could be implemented in such a way, that it would support both standards. However, due to less strict timing issues and less complexity the focus is on the MMC protocol, which is downward compatible to the SD standard.

Following the assignment as MMC card, some features like supply voltage, maximum speed, number of communication lines on the SD bus, capacity of the card, etc. are requested by the SD card reader. Due to several issues (see Section IV) the supply voltage is determined as 3.3 V, the maximum speed of the emulated card is set to 400 kHz, the number of communication lines is set to one and the capacity of the card is determined as 128 MBytes. These setup values are processed within the so called *Card Identification mode*

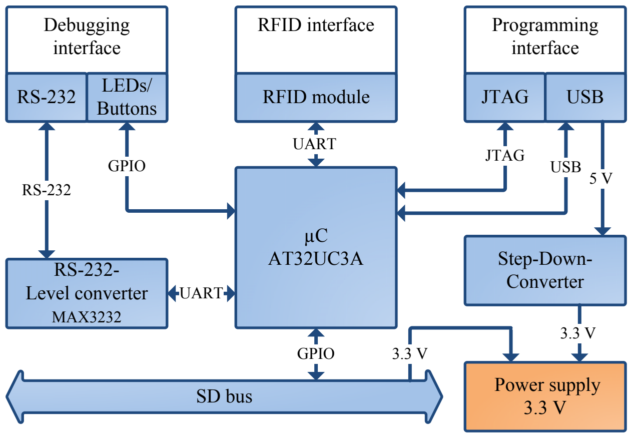


Figure 2. Block diagram of the reader's hardware

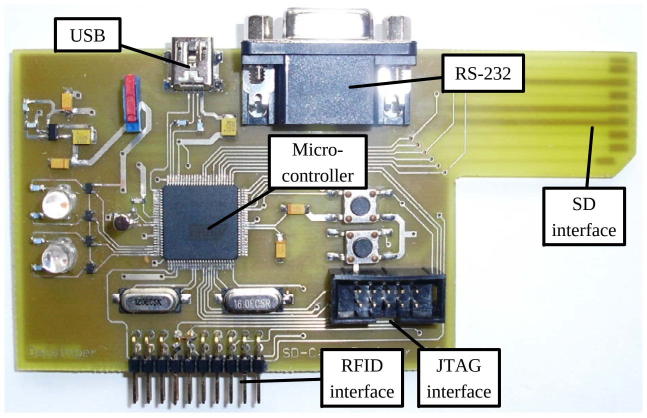


Figure 3. The prototype of the SD-capable RFID reader, shown without the external RFID reader module, which is usually connected to the RFID interface

running at low speed (max. 400 kHz). Subsequently, the card enters the *Data Transfer mode* making the card's content available, e.g., to the OS. Within this data-transfer mode, every time the SD card reader requests data information, a data block of 512 Bytes needs to be transferred from the MMC card, which is emulated by the microcontroller, to the card reader and vice versa. This project uses the FAT16 architecture to implement a particular memory structure. FAT16 is used because of its prevalence in almost all computer systems. Therefore, the microcontroller has to emulate a FAT16 formatted memory architecture containing Master Boot Record (MBR), Volume Boot Record (VBR), and the root directory. The SD card reader, finally, provides the application system, usually a PC, PDA or smartphone, with the emulated card's data, i.e., files and directories. Currently, there are four files (FAT.HEX, ROOTDIR.HEX, VBR.HEX and RFID.TXT) with the latter file containing the information (in this case the UIDs) of the RFID transponders (see Figure 6). The other three files contain the structure of the emulated FAT, VBR, and root directory.

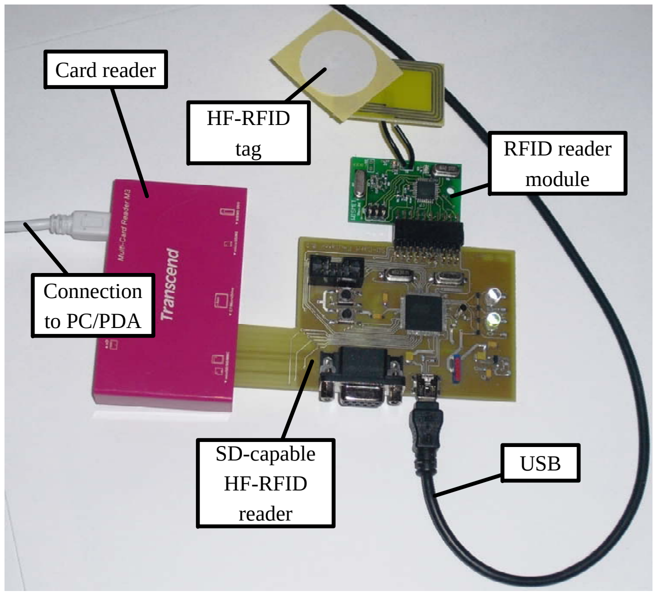


Figure 5. The system setup for verification

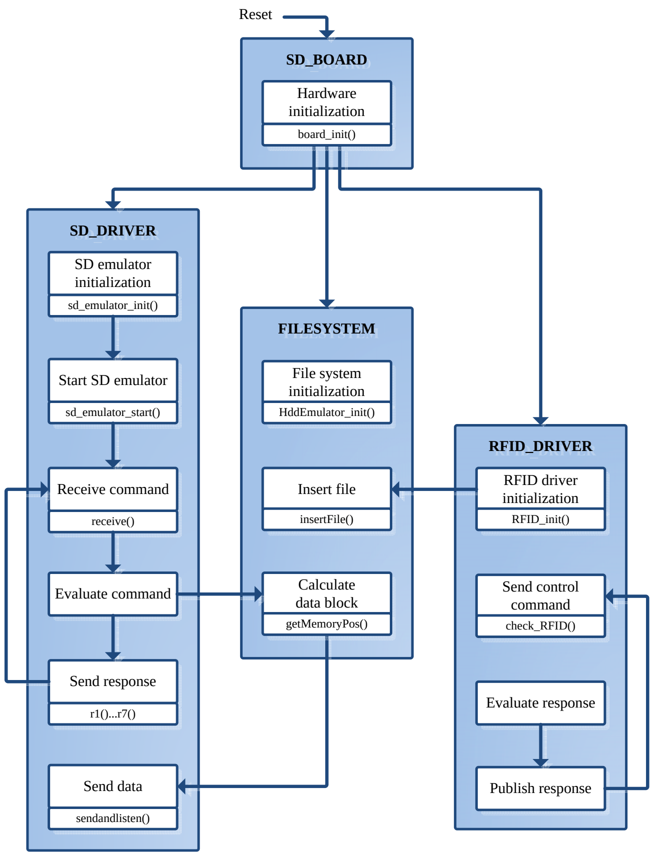


Figure 4. Overview of the SD-capable HF-RFID reader's firmware

*2) Realization of the SD Protocol within the Firmware of the Reader:* Figure 4 shows a rough overview of the system's firmware structure and the appropriate work flow. After starting the µC by applying power, the system initializes its hardware parts (Reset). Subsequently, three main parts are loaded, running in three different threads within a thread-like environment. In this context, *thread-like* describes an environment, in which threads are treated separately but consecutively in an quasi-OS (operating system) environment as the µC includes a single-core CPU. The *RFID_DRIVER* part (right hand side in Figure 4) has two main functionalities. The first functionality covers the control (initialization and communication) of the RFID module. This includes, e.g., the request for the available transponders, with the resulting and received UIDs, respectively. The second functionality is the preparation of the transponders' data for the adoption to the FAT file system.

Taking a more detailed view at the reader's firmware shows, that there is one major problem regarding the timing issues of the SD protocol. This already above mentioned problem is further discussed in Section IV.

The part *FILESYSTEM* (centered in Figure 4) has three main functionalities. The first one involves the generation of the underlying FAT16 file system. The file system is initialized by creating its administrative datasets: MBR, VBR, and root directory. The communication with the RFID part (*RFID_DRIVER*) and the SD part (*SD_DRIVER*) describes the second functionality of the *FILESYSTEM* part. The third functionality is responsible for creating the appropriate files within the FAT16 file system. Additionally, the content of the files has to be created, too, including the transponders' data (see Figure 6).

The *SD_DRIVER* part (left hand side in Figure 4) has to deal with the SD respectively MMC protocol. This means, all timing issues regarding the SD bus have to be handled by this part of the firmware. The part is µC hardware dependent, and therefore the system's performance is somehow limited at this point. The addressed hardware dependence is directly related to the hardware components, particularly the µC with its rather limited capabilities to cope with fast incoming and outgoing signals. This issue is further discussed in Section IV. However, the realization of this part of the firmware is mainly done by polling several internal flags of the µC. The reason for not choosing interrupts is caused by the overwhelming amount of stack an interrupt-driven approach would need. The main drawback would not be the memory usage but the loss of time needed to return from the sub-functions (e.g., interrupt routines). The main external signal, and therefore the triggered internal flag of the µC, is the SD clock signal generated by the SD card reader. By choosing an interrupt-driven approach, the time for calling the interrupt function and returning from it (to the main function) would not fulfill the needs for an appropriate operation of the system.

By briefly summarizing the firmware architecture, one can say that the firmware controls the external RFID module to get the transponders' data (e.g., the UIDs). This data is evaluated and packed into the emulated FAT16 structure of the system. A connected SD card reader may read the FAT16 structure and forward the data to the overlying operating system, which can display the appropriate data (UID) within a file.

## III. Verification

This section describes the verification of the SD-RFID combination. The setup to verify the system is given as in Figure 5. The figure shows the SD card reader, which is connected to a computer via USB (left side of the figure), whereas the SD-capable HF-RFID reader is located in the center. The RFID reader module (with antenna and RFID tag) is connected at the upper side of the reader. A USB debugging connection is shown at the bottom of the picture. The system is connected to the SD card reader (Type: Transcend Multi-Card Reader M3).

To prove the correct operation of the system, the SD card reader and the system were connected to different operating systems, including Windows XP, Windows 7, Linux (Ubuntu) and Mac OS X (Snow Leopard). The results are shown in Figure 6. The screenshot on the left was made in Windows 7 showing additionally the four different files, including the *RFID.txt* containing the UID of the transponder. The UID with the value of '00 00 00 00' stands for an invalid data communication between transponder and reader. The current UID of the transponder used is therefore '3C 50 8B 2A'. The Linux screenshot is located on the bottom of the picture and shows the capacity of the card; in this case it is 128 MByte. The background of the picture shows a Mac OS X screenshot containing the content of the emulated FAT16 system (top left), the content of the *RFID.txt* file (center right), and the information of the SD-capable HF-RFID reader, called *EMULATOR* (right side of the figure).

## IV. Performance Issues

The limitations appearing during implementation are mostly generated by various SD card reader implementations of the SD protocol standard. A huge drawback is the minimum clock rate, various card readers manage. The maximum frequency for initializing the SD card (400 kHz) is managed and adhered by every card reader.

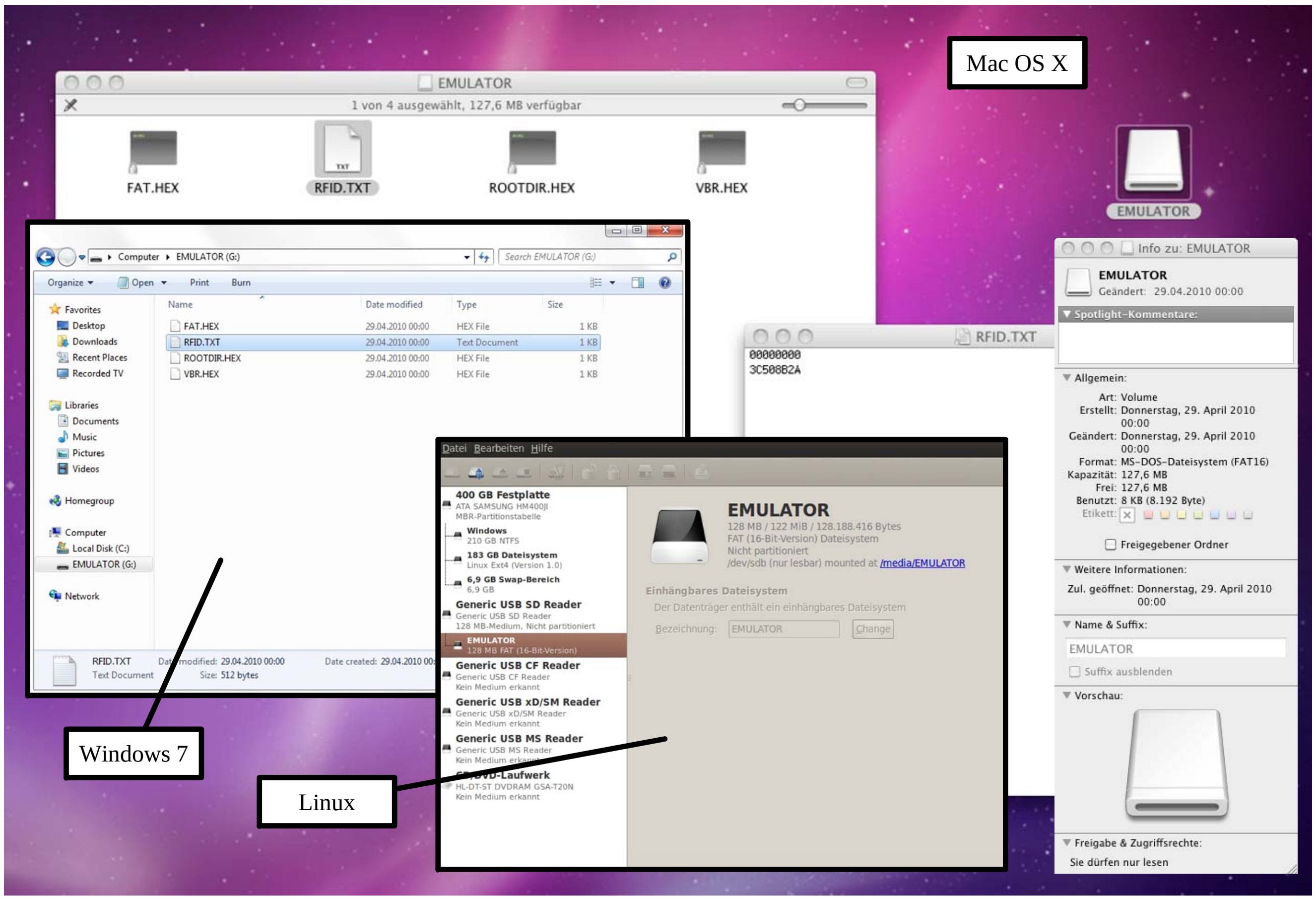
Although the SD card (or MMC card) can return its maximum allowed clock rate to the SD card reader to prevent the reader to read out data too fast, some SD card readers ignore that property and start off with a frequency far too high for the µC to cope with. Other card readers just reset and search for another card. These problems create some kind of drawback for this particular system.

The limitations itself are governed by the maximum clock rate the microcontroller is able to handle. Internal calculations showed that the maximum clock rate is about 1 MHz using the µC at the maximum clock rate of 66 MHz. These timing issues are the reason why a flag-driven polling approach is used instead of choosing an interrupt-driven approach. It can be shown that polling the flags (which are also used by the interrupts) provides a higher data throughput than an interrupt-driven approach. The bottleneck, regarding the timing problem, occurred at the point where SD data blocks of 512 Byte have to be transferred from the card reader to the system and vice versa, as some processes have to work in parallel, which, of course takes time, if only a single-core CPU is available.

## V. Conclusion and Future Work

This paper presented an approach to overcome existing obstacles with RFID reader architectures using the SD interface. While standard SD reader approaches generally use the SDIO interface, the work presented in this paper shows a method to use the standard SD interface to work with every possible kind of SD card reader. The system is based on a microcontroller fully emulating a FAT16 file system to be able to transfer RFID-based data, e.g., the UID of a transponder, to a superior system, e.g., a PC, PDA or smartphone, using a standard SD card reader interface. Therefore, the system not only controls a connected RFID reader module, but also the SD or MMC card protocol to communicate with the SD card reader. Successful tests with various operating systems have been carried out (see Figure 6) to prove the principle of this structure.

During the work different kinds of limitations were located. Future implementations should account for these drawbacks by implementing time robust structures. For instance, one option would inherit FPGA structures for the direct interface to the SD card reader. Also, other options



EMULATOR

1 von 4 ausgewählt, 127,6 MB verfügbar

Computer ▸ EMULATOR (G:)

Search EMULATOR (G:)

| Name | Date modified | Type | Size |
|---|---|---|---|
| FAT.HEX | 29.04.2010 00:00 | HEX File | 1 KB |
| RFID.TXT | 29.04.2010 00:00 | Text Document | 1 KB |
| ROOTDIR.HEX | 29.04.2010 00:00 | HEX File | 1 KB |
| VBR.HEX | 29.04.2010 00:00 | HEX File | 1 KB |

RFID.TXT Date modified: 29.04.2010 00:00 Date created: 29.04.2010 00:
Text Document Size: 512 bytes

RFID.TXT

```
00000000
3C508B2A
```

400 GB Festplatte
ATA SAMSUNG HM400JI
MBR-Partitionstabelle

Windows
210 GB NTFS

183 GB Dateisystem
Linux Ext4 (Version 1.0)

6,9 GB Swap-Bereich
6,9 GB

Generic USB SD Reader
Generic USB SD Reader
128 MB-Medium, Nicht partitioniert

EMULATOR
128 MB FAT (16-Bit-Version)

Generic USB CF Reader
Generic USB CF Reader
Kein Medium erkannt

Generic USB xD/SM Reader
Generic USB xD/SM Reader
Kein Medium erkannt

Generic USB MS Reader
Generic USB MS Reader
Kein Medium erkannt

CD/DVD-Laufwerk
HL-DT-ST DVDRAM GSA-T20N
Kein Medium erkannt

EMULATOR
128 MB / 122 MiB / 128.188.416 Bytes
FAT (16-Bit-Version) Dateisystem
Nicht partitioniert
/dev/sdb (nur lesbar) mounted at /media/EMULATOR

Einhängbares Dateisystem
Der Datenträger enthält ein einhängbares Dateisystem
Bezeichnung: EMULATOR

Info zu: EMULATOR

EMULATOR
Geändert: 29.04.2010 00:00

▼ Spotlight-Kommentare:

▼ Allgemein:

Art: Volume
Erstellt: Donnerstag, 29. April 2010 00:00
Geändert: Donnerstag, 29. April 2010 00:00
Format: MS-DOS-Dateisystem (FAT16)
Kapazität: 127,6 MB
Frei: 127,6 MB
Benutzt: 8 KB (8.192 Byte)
Etikett:

☐ Freigegebener Ordner

▼ Weitere Informationen:

Zul. geöffnet: Donnerstag, 29. April 2010 00:00

▼ Name & Suffix:

EMULATOR
☐ Suffix ausblenden

▼ Vorschau:

▼ Freigabe & Zugriffsrechte:

Sie dürfen nur lesen

Figure 6. Verification of the system with different operating systems: Linux (Ubuntu 9.04, [10]), Windows 7 [11] and Mac OS X (Snow Leopard, [12])

like CPLDs (Complex Programmable Logic Device) would come to the fore.

## Acknowledgment

The authors would like to thank the Fraunhofer Institute for Integrated Circuits (IIS) in Erlangen for its support on this project. Additionally, we would like to thank Heinrich Dietsch for useful discussions and the anonymous reviewers for providing helpful comments.

## References

[1] trading-house.net AG, "Research - RFID Market to Reach $5.35 Billion This Year, Says ABI Research," *www.ad-hoc-news.de*, Mar 2010. [Online]. Available: http://www.ad-hoc-news.de/research-rfid-market-to-reach-5-35-billion-this--/de/Unternehmensnachrichten/21106631

[2] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. Wiley, 2010.

[3] A. Loeffler, U. Wissendheit, H. Gerhaeuser, M. Hoffmann, A. M. Zadeh, and D. Kuznetsova, "A SOAP capable HF-RFID-Reader," in *RFID SysTech 2008 , 4th European Workshop on RFID Systems and Technologies*, 2008.

[4] SD Association, "http://www.sdcard.org/home/," Feb 2010. [Online]. Available: {http://www.sdcard.org/home/}

[5] RFReader Corporation, "http://www.rfreader.com/," Mar 2010. [Online]. Available: {http://www.rfreader.com/}

[6] Sirit Inc., "http://www.sirit.com/," Mar 2010. [Online]. Available: {http://www.sirit.com/}

[7] J. Axelson, *USB mass storage: designing and programming devices and embedded hosts*, ser. E-libro. Lakeview Research LLC, 2006.

[8] Atmel Corporation, "http://www.atmel.com/," Jan 2010. [Online]. Available: {http://www.atmel.com/}

[9] MultiMediaCard Association, "http://www.mmca.org/," Feb 2010. [Online]. Available: {http://www.mmca.org/}

[10] R. Petersen, *Ubuntu 9.04 Desktop Handbook*. Surfing Turtle Press, 2009.

[11] J. Boyce, *Windows 7 Bible*, ser. Bible Series. John Wiley & Sons, 2009.

[12] D. Pogue, *Mac OS X Snow Leopard: The Missing Manual*, ser. Missing manual. O'Reilly Media, 2009.